SilverLine: Data and Network Isolation for Cloud Services

Yogesh Mundada, Anirudh Ramachandran, and Nick Feamster
School of Computer Science, Georgia Tech

ABSTRACT

Although cloud computing service providers offer opportunities for improving the administration, reliability, and maintenance of hosted services, they also concentrate network resources and data in a small number of cloud service providers. The concentration of data and resources introduces various associated risks, including sharing the underlying infrastructure with unknown (and untrusted) tenants and relying on the availability and security of the underlying infrastructure itself. These security risks represent some of the most significant barriers to the adoption of cloud-based services. To begin tackling these risks, a cloud hosting infrastructure should provide strong guarantees for resource and data isolation. This paper examines isolation problems with today's cloud hosting infrastructures and proposes SilverLine, a collection of techniques to improve data and network isolation for a cloud tenant's service.

1. Introduction

Cloud computing reduces operational costs for networked and Web-based services and makes scaling easier as the demands for hosted services grow. Additionally, cloud computing lowers the barrier to entry for new enterprises and services by allowing them to direct energy towards the development of new functions, as opposed to low-level deployment or system administration concerns. Indeed, Gartner reports that revenue from cloud services was $68.3 billion in 2010 and projects this figure to reach nearly $150 billion by 2014 [16].

Although cloud hosting continues to become more prominent, many enterprises are hesitant to deploy services in the cloud. Indeed, the very nature of cloud computing that has propelled its rise—low barrier-to-entry and third-party management of infrastructure and system administration—has also become a significant concern. The Cloud Security Alliance indicates that potentially improper data and network *isolation* is a top threat for medium-to-large enterprises [15]. Any attacker with a stolen credit card number can purchase their own cloud services, thus lowering the barrier to entry for attackers. The lack of control over management is exacerbated because many tenants share the same underlying physical infrastructure, making it potentially easier for a malicious tenant to mount an attack (*e.g.*, a denial of service) against another. Additional threats—either from another cloud tenant or an outside attacker—exist because cloud services are extremely easy to deploy, especially with platforms such as Google's AppEngine [17] or Heroku [18]: many Web services are created by novice programmers without proper code security audits. As a result of these threats, cloud providers have a strong incentive to deploy and maintain technologies that help improve the security of cloud services.

Although corporations and individuals are entrusting an increasing amount of personal and corporate information in cloud solutions (*e.g.*, Dropbox), relatively little attention has been paid to the security of data that is stored in the cloud. If an external attacker or a malicious cloud tenant gains access to a vulnerable or misconfigured service of another tenant, current techniques—including content-based data loss prevention systems [11]—cannot stop them froma gaining access to data associated with the service. In one recent example, Microsoft's cloud-hosted Business Productivity Online Suite suffered a misconfiguration that allowed unauthorized users to download other tenant data [19].

This paper introduces SilverLine, a system that enables cloud providers to offer *security as a service* to protect tenant data in clouds, even if the software or services that a particular tenant runs are themselves insecure. SilverLine augments the cloud provider's virtual machine manager (*e.g.*, XenServer, VMware ESXi, etc.) to perform data and network isolation between various cloud tenants, and combines this isolation with modifications to guest operating systems for additional data isolation. These mechanisms protect against data leaks that result from compromise, misconfiguration, or side-channel attacks from co-resident cloud tenants [10].

SilverLine implements two types of isolation: (1) *data isolation* and (2) *network isolation*. To enforce data isolation, we use a transparent operating system-level information-flow tracking layer that is assisted by an enforcement layer in the cloud provider's virtual machine monitor (*e.g.*, Xen or VMware). Our approach allows tenants to label data with security levels; trusted enforcers then use these labels to ensure that data from one tenant is not propagated to untrusted server instances belonging to other tenants, or to locations outside the cloud. Even in case of a service misconfiguration or exploit that would otherwise have resulted in a breach or unauthorized access of data, labels that are tracked and enforced by the cloud provider's infrastructure would ensure that such breaches would not occur

with SilverLine. To improve network isolation, we develop defenses against *co-residence checks* that allow an attacker to identify a victim tenant's virtual machine instances. SilverLine runs entirely in software (*i.e.*, without customized hardware) using existing network infrastructure and emerging network control protocols (*e.g.*, OpenFlow [8], which can already be deployed on virtual switches in the Xen hypervisor that is used by Amazon's AWS).

2. Background and Related Work

Cloud Security. Cloud-based infrastructures face many threats. First, existing cloud providers' default security mechanisms still place much of the onus of data and resource isolation on the cloud tenants themselves [14]; because the tenant is typically free to choose the OS and services that run on their virtual machines, their services become vulnerable to attacks against the OS and services both from within and outside the cloud. Second, most cloud service developers are not security experts, and misconfigurations, buggy code, and weak passwords expose their services to critical data breaches, such as the Microsoft Business Productivity Online Suite's recent misconfiguration [19], and the breach of its corporate emails from HBGary—itself a security firm [4]. Finally, cloud tenants do not control the physical machines that host their virtual machine instances, and a malicious tenant or even an insider (*e.g.*, an operator of the cloud platform itself) might be able to affect or breach the security of a co-resident virtual machine using side-channel attacks [10].

In 2009, Ristenpart *et al.* studied how cloud hosting providers allocate IP addresses to virtual machines. With knowledge of this allocation strategy, they identified a set of possible physical machines where the victim's virtual machine might be running; an attacker could use this technique to increase the likelihood of running a VM instance on the same physical machine as the victim and affecting a DoS attack [10]. They also design a *co-residence* check, which is a side-channel attack that allows an attacker to verify whether his VM is running on the same physical hardware as the victim, using network-based (*e.g.*, based on IP address allocation, traceroutes, or latencies between machines) or cache-based measurements. In Section 4.2, we describe how SilverLine can prevent certain types of co-residence checks.

Hao *et al.* describe a virtualized network infrastructure to isolate tenants from one another and defend against co-residence checks [3]. Unfortunately, their design requires specialized hardware devices at the boundaries of edge domains; two tenants who want to use each other's software services cannot do so without significant administrative and configuration overhead. The design also tracks at least four mappings at the central controller and uses VLANs to isolate traffic between two tenants within each edge domain, which can be complex. Neither solution proposes the type of data isolation offered by SilverLine. Recent work has proposed extending trusted computing platforms [1] to the cloud by providing protocols to launch and migrate virtual machines over trusted platforms [12]. These methods protect against certain types of attacks by the cloud infrastructure provider but do not defend against service misconfiguration and exploits or attacks by other malicious tenants.

Information Flow Control. SilverLine's data isolation capabilities are inspired by *information-flow control* mechanisms in operating systems such as DStar [20] and Flume [5]. The main drawback of these systems is that they prevent a set of multi-process applications (including one or more untrusted applications) from leaking data, so they require applications to be rewritten. In contrast, SilverLine does not require applications to be rewritten; rather, it tracks information flow in a modified Linux kernel. It also provides services for tenants to maintain and enforce their information flow policies. Other systems such as Neon [21] track information flow for legacy applications entirely within the hypervisor and do instruction level taint tracking within the emulator. It incurs a higher overhead as execution control is required to be passed to QEMU instance within the root domain (dom0). Neon thus achieves finer granularity, at the cost of more performance overhead.

3. Threat Model

Table 1 shows the types of threats that we consider, the entities that pose each threat, and defenses against each type of attack. The most likely threat for cloud tenants arises from misconfigurations or bugs in the tenant's own service. Unfortunately, not only must the tenant secure their Internet-facing services, but also it must ensure that a malicious tenant hosted on the same cloud provider cannot disrupt or steal information from the tenant's virtual machine instances.

Other threats include attacks due to vulnerabilities in the cloud platform itself (*e.g.*, the Amazon EC2 attack [10]), vulnerabilities in the operating system run by a tenant, or breaches due to malicious insiders employed by the cloud service provider. SilverLine currently trusts a tenant's operating system kernel to perform information-flow tracking. Critical operating system bugs and vulnerabilities are not common, but, to protect against such threats, the cloud service provider could deploy an information flow tracking system similar to SilverLine in the *hypervisor*. SilverLine cannot protect against attacks due to vulnerabilities in the cloud platform or attacks from malicious insiders, but recent work on trusted cloud computing platforms [12] that builds upon Trusted Platform Modules (TPM) may help

Threat	Victim	Attacker	Likelihood	Consequences	Solutions
A tenant's deployed service is misconfigured or vulnerable	Only the tenant	Other tenants and outsiders	Most likely	Loss of data & service disruption	Bug-free code, Firewalls & IDSes, *SilverLine* (Dom0 + OS)
Cloud platform allows side-channel attacks	All cloud tenants	Other tenants	Less likely	Denial of Service and potential data breach	Trusted Cloud Platforms [12], *SilverLine* in VMM
A tenant's system libraries, kernel has bugs	Only the tenant	Outsiders or Other cloud tenants	Less likely	Loss of data & service disruption	Timely security updates, *SilverLine* in the VMM
Cloud provider is malicious	All cloud tenants	Cloud provider's employees	Least	Data and service breach for all tenants	Background checks, full encryption

Table 1: Taxonomy of threats specific to cloud computing.

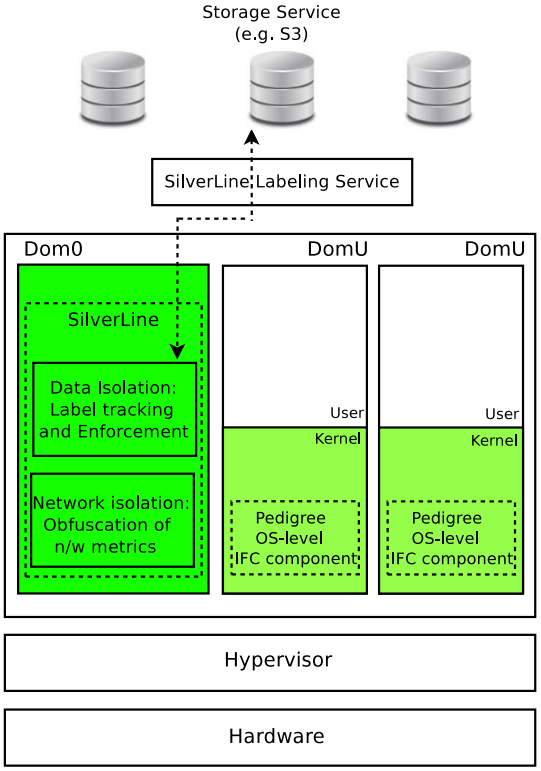


Figure 1: SilverLine is installed by the Cloud service provider in the privileged guest OS (*e.g.*, Dom0 in Xen), and an OS-level information-flow control system, Pedigree, is installed in guest OSes. Darker shades of green indicate higher trustworthiness.

mitigate some of these attacks.

4. SilverLine: Isolation for Cloud Services

Figure 1 shows how SilverLine resides on the virtualized hardware that cloud tenants use in a cloud hosting environment. The design has three parts: (1) two software modules in the privileged guest OS (dom0), one each for data isolation and network isolation; (2) a labeling service attached to the cloud provider's storage services (*e.g.*, a database service such as Amazon's Relational Database Service); and (3) an OS-level information-flow tracking component called *Pedigree*, which is installed on virtual machine instances. Cloud tenants that run Pedigree can specify policies that will automatically assign *labels* to their data using the labeling service attached to the cloud provider's storage services. Pedigree then tracks the flow of information using labels between all processes and files associated with the tenant's VM instances. If the tenant's data accidentally travels to another tenant's VM or to a machine outside the cloud provider's network, the *enforcer* component in the dom0 will stop such potential data breaches. The network isolation component in each dom0 obfuscates co-residency checks: using a centralized database or service, it rewrites IP address ranges to thwart attackers from singling out a victim tenant's VM; it also normalizes ping times such that the ping times between VMs on the same physical machine are no different than ping times between VMs on different machines. We describe data isolation in Section 4.1, and network isolation in Section 4.2.

4.1 Data Isolation

We discuss the three components of SilverLine that enable data isolation.

Pedigree Information-flow Tracking. Pedigree uses *labels* to track information flow between files and process within a single machine as well as across the network. Using a trusted module in a Linux kernel, Pedigree provides network-wide information flow control for unmodified legacy applications [6]. Users or administrators add labels to files and use *enforcers* on both end-hosts and at a network egress point to prevent data leaks. Pedigree's labels have associated policies that specify the users who can access and tagged with the label, and who can remove labels attached to data ("declassification"). Pedigree monitors interactions between resources (such as processes or files) and tracks information flow using labels associated with every resource, and prevents data leaks from the entire enterprise network (instead of a single application) by preventing users or applications from exporting data that carries a user's label outside the network boundary. In a SilverLine-enabled cloud, a tenant can choose to install a Pedigree-enabled VM image on all its VM instances to benefit from the provider's security services.

Labeling Service. The labeling service automatically

labels data inserted into the storage service, and ensures that data returned for read requests have these labels associated with them. Tenants can create policies and labels through a management console. The service functions on top of existing infrastructure such as Amazon's RDS, SimpleDB, or even its block-based storage service S3 [13]. A simple policy description and an associated label that is generated and attached to a database record might look as follows:

when query := "INSERT" **and** table := "USERS":
Generate new label; add it to the DB record

Enforcer. The enforcer exists in all dom0s on the cloud and prevents data flow both between tenants on the came infrastructure and between the cloud infrastructure and the global Internet.

Preventing unauthorized data flow between tenants that share a cloud infrastructure is straightforward: Using the unique tenant-id field in labels associated with data generated by a specific tenant, the cloud provider can ensure that any data transfer that originates from the tenant's VM or storage carries that label. The enforcer in the dom0 on each machine in the cloud infrastructure will intercept and read the tenant-id label on each connection, and ensure that the destination IP address belongs to the same tenant. Thus even if a tenant misconfigures its services such that its files are accessible to any tenant, other tenants (or their users) will not be able to access the victim tenant's data due to the tenant-id label check.

Preventing data breaches from the cloud hosting infrastructure to external networks, however, is considerably more challenging. The challenge is to allow authenticated users to view their own data from outside the cloud while preventing an attacker from viewing any other users' data, even if the attacker has exploited a vulnerability in Internet-facing services of the cloud tenant. For this, the cloud tenant runs a trusted login service that is coupled with the cloud provider's declassifier service.

Each tenant implements their own *trusted* login service, but ensures that it is free of vulnerabilities. To do so, they might use an audited open-source library and deploy the login service on a VM of its own. When a user enters their credentials at the tenant's Website, the login process authenticates the user and communicates with the declassifier to indicate the connection that belongs to the user.

The declassifier is a *trusted* proxy service (which we expect to be provided by the cloud service provider) that can strip labels from flows before forwarding the traffic. When the declassifier receives a request from a tenant's login service to declassify a particular user's connection, it will ensure that any outgoing labeled data on that connection that has the correct tenant-id and user; before forwarding the traffic, the declassifier will remove

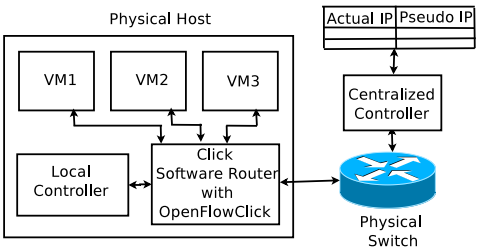


Figure 2: Rewriting internal IP addresses as pseudo-IP addresses.

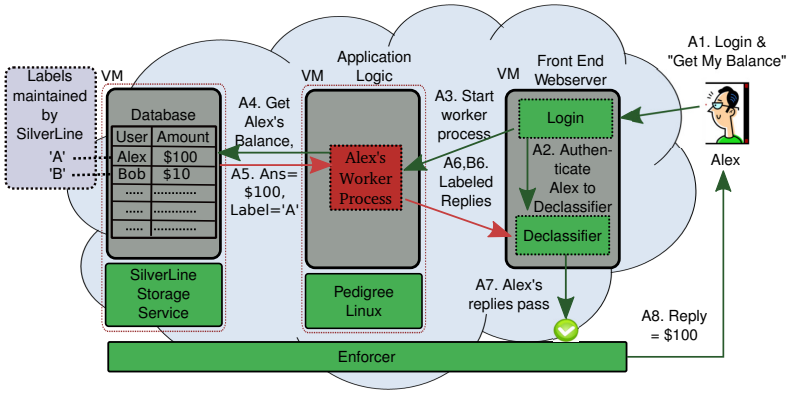
the associated labels. Outgoing data is always routed through a network enforcer—which can be a standalone router or switch that can inspect labels—which has one simple function: if it observes an outgoing flow that still has a label attached to its data, the flow will be dropped.

Figure 2(a) and Figure 2(b) show how SilverLine enforces data isolation between the cloud and external users. Suppose that a tenant deploys a Web service that accepts user logins and displays user-specific information (*e.g.*, an online banking site). All of the tenant's virtual machines run Pedigree, all dom0s run SilverLine enforcer and access the data from SilverLine storage service VM. The tenant specifies SilverLine policies so that each user's data receives a unique label. After a successful login, the login process passes the connection-id and the username to the declassifier. New worker threads are spawned for each logged-in user that read *labeled* user data from the storage service and output the same data *but with the label removed*. The output data is sent from the declassifier successfully flows through the enforcer, because the data has already been declassified.

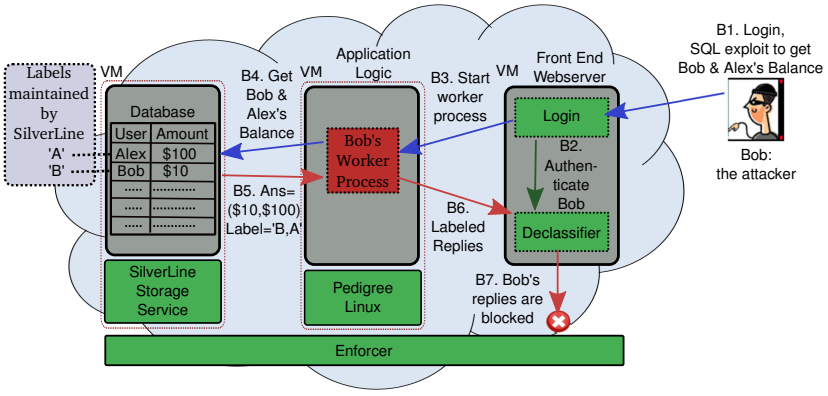
Figure 2(a) also shows the interactions between a normal user Alex and SilverLine. Alex logs in the system and requests for his balance. His responses are labeled only with his label 'A'. The then declassifier removes Alex's label; because the enforcer does not find any sensitive labels attached to the replies, forwards the traffic. On the other hand, lets say a malicious user Bob exploits an SQL injection bug in the Internet-facing Web service of the tenant and attempts to retrieve Alex's data as shown in Figure 2(b). Since the declassifier has only associated Bob's connection with Bob's label, it will only strip Bob's label from database replies. Alex's data that Bob requested will continue to have Alex's label associated with it when it reaches the enforcer, and the enforcer will drop such packets. Even if Bob can trick any of the other internal VMs into sending the data directly outside bypassing the declassifier VM, the enforcers in each of the dom0s will drop that connection.

4.2 Network Isolation

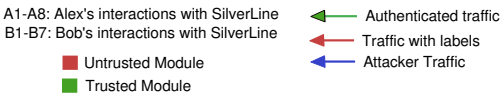
A malicious cloud tenant can use side-channel attacks based on network-based and cache-based channels to gain co-residence on a victim tenant's VM [10]. To isolate the victim's VMs in a cloud infrastructure, Risten-



(a) SilverLine operation: Normal user



(b) SilverLine operation: Attacker



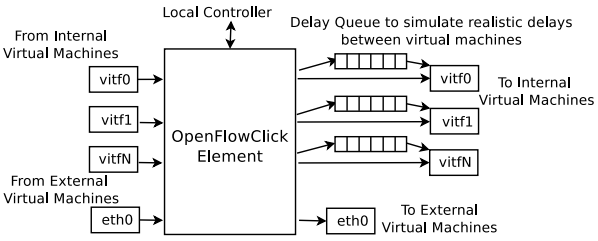


Figure 3: Normalizing round-trip times.

part *et al.* rely on discovering the internal cloud infrastructure. We use the OpenFlow/NOX platform [2, 7]—already included in open-source cloud platforms such as the Xen Cloud Platform (XCP)—to perform the following defenses.

Pseudo IP addresses for virtual machines. Because Ristenpart *et al.* used IP address ranges to single out the physical machines running a victim's VM, we aim to provide each VM with a "pseudo" randomly-allocated IP address that VMs use when communicating with each other, keeping the actual cloud-provider IP address allocations unchanged. The SilverLine network isolation module in each dom0 includes an OpenFlow switch and a NOX controller (*e.g.*, using OpenflowSwitch [8]) that rewrites pseudo IP addresses to actual VM IP address before packets leave the machine. The mapping between pseudo IP addresses and the virtual machine's actual IP addresses maintained by a single centralized service or database; local NOX controllers on each physical machine consult the centralized database for these mappings as shown in Figure 2. Because internal IP addresses are discovered using DNS requests, the local controller also rewrites DNS responses to the appropriate pseudo address. ARP queries and responses also need to be rewritten, and hence they are redirected to local NOX controller as well. Because pseudo IP addresses are only used within virtual machines, switches and routers in the cloud provider's network can continue to forward packets based on their statically assigned IP addresses.

Normalized network metrics. Ristenpart *et al.* observed that the round trip times (RTTs) between virtual machines running on the same physical machine are significantly lower than RTTs between VMs running on

separate physical machines [10]. SilverLine normalizes round-trip delays by adding a small queuing delay to packets sent between virtual machines that reside on the same physical machine. To do so, SilverLine uses an OpenFlow element we developed for the Click software router [9], along with a queuing element as shown in Figure 3. Another co-residence check involves an attacker running a traceroute to determine hop count to the victim; if this hop count is zero or one, then the attacker is likely co-resident on the victim's virtual machine. To defend against this attack, for traffic between virtual machines within a physical machine, SilverLine can drop packets with low TTL values.

5. Summary and Research Agenda

One of the biggest impediments to the adoption of cloud-based services is the risk that data stored in the cloud will be compromised, either by a third-party attacker or by another cloud tenant. To prevent these emerging threats, we have designed SilverLine, a system that improves both data and network isolation for cloud-based services without requiring cloud application developers to rewrite their applications, and without requiring the deployment of specialized network hardware. SilverLine provides isolation for cloud services using a specialized kernel module and an addition to the privileged guest OS of the cloud provider's virtual machine monitor.

Our existing implementation of Pedigree, which performs information-flow control in enterprise networks, suggests that the performance overhead for propagating labels incurs acceptable performance overhead, and our implementation of the `OpenFlowClick` [9] module suggests that the performance penalty for pseudo-IP mapping and network normalization is minimal. Our next step is to deploy SilverLine in a cloud environment and test the performance overhead for cloud-based services. We also intend to further develop SilverLine to defend against covert channels and insider threats (*i.e.*, attacks by administrators of the cloud infrastructure). Finally, we intend to extend SilverLine to track labels at a finer granularity than files and processes.

Acknowledgments

This research was supported by NSF Awards CNS-0916732 and CNS-0964647.

REFERENCES

[1] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Bone. Terra: A Virtual Machine-Based Platform for Trusted Computing. In *Proc. 19th ACM Symposium on Operating Systems Principles (SOSP)*, Lake George, NY, Oct. 2003.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, July 2008.

[3] F. Hao, T. Lakshman, S. Mukherjee, and H. Song. Secure Cloud Computing with a Virtualized Network Infrastructure. Boston, MA, June 2010.

[4] Anonymous speaks: the inside story of the hbgary hack. `http://arstechnica.com/tech-policy/news/2011/02/anonymous-speaks-the-inside-story-of-the-hbgary-hack.ars/`.

[5] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris. Information flow control for standard os abstractions. In *Proc. 21st ACM Symposium on Operating Systems Principles (SOSP)*, Stevenson, WA, Oct. 2007.

[6] Y. Mundada, A. Ramachandran, M. B. Tariq, and N. Feamster. Practical Data-Leak Prevention for Legacy Applications in Enterprise Networks. Technical Report GT-CS-11-01, Georgia Institute of Technology, Atlanta, GA, Jan. 2011.

[7] NOX: An OpenFlow controller. `http://www.noxrepo.org`.

[8] OpenFlow Switch Consortium. `http://www.openflowswitch.org/`, 2008.

[9] OpenFlow Click element. `http://www.openflowswitch.org/wk/index.php/OpenFlowClick`, 2009.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud! exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, 2009.

[11] RSA Data Loss Prevention. `http://www.rsa.com/node.aspx?id=3426`.

[12] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards Trusted Cloud Computing. San Diego, CA, June 2009.

[13] Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3/`.

[14] Amazon Web Services: Overview of Security Processes (section Port Scanning). `http://awsmedia.s3.amazonaws.com/pdf/AWS_Security_Whitepaper.pdf`.

[15] Cloud Security Alliance's Top Threats to Cloud Computing. `http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`.

[16] Cloud Services To Top $68 Billion In 2010. `http://www.informationweek.com/news/storage/systems/showArticle.jhtml?articleID=225701068`.

[17] Google App Engine. `http://code.google.com/appengine/`.

[18] Heroku — Ruby Cloud Platform as a Service. `http://www.heroku.com/`.

[19] Microsoft BPOS Cloud Service Hit With Data Breach. `http://www.pcworld.com/businesscenter/article/214591/microsoft_bpos_cloud_service_hit_with_data_breach.html`.

[20] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazieres. Making Information Flow Explicit in HiStar. In *Proc. 7th USENIX OSDI*, Seattle, WA, Nov. 2006.

[21] Q. Zhang, J. McCullough, J. Ma, N. Schear, M. Vrable, A. C. Snoeren, G. M. Voelker, S. Savage, and A. Vahdat. Neon: System Support for Derived Data Management. In *ACM Conference on Virtual Execution Environments*, Mar. 2010.